ÖREBRO UNIVERSITET

INSTITUTIONEN FÖR
NATURVETENSKAP OCH TEKNIK

Kursens namn	**Inbyggda system för civilingenjörer – DT511G**
Examinationsmomentets namn/provkod	**0100**
Datum	2019-03-25
Tid	Kl. 14:15 – 17:15

Tillåtna hjälpmedel	Calculator
Instruktion	Read all the questions carefully. Start answering each question in a new answer sheet. Write only on one side of an answer sheet. Write the exam code on every answer sheet. Write in a readable way!
Viktigt att tänka på	
Ansvarig/-a lärare (ev. telefonnummer)	Farhang Nemati Tel: 019-303264 Mobil: 0702533418
Totalt antal poäng	40
Betyg (ev. ECTS)	The exam contains 5 questions. The total points are 40. At least 20 points are required for grade 3 (pass), 30 points for grade 4 and 35 points for grade 5
Tentamensresultat	The results will be notified in Studentforum within 15 working days after the exam.
Övrigt	You can write your answers in English or Swedish.

Good Luck!

1. (10 points)

Briefly answer the following questions (You may answer a question by an example or a drawing if it makes sense):

a) How tasks execute in a “Monolithic” way?
b) What is Cross-Platform Development?
c) Mention two advantages of “concurrency” in embedded systems.
d) What is an atomic operation?
e) Mention two facilities that a Real-Time Operating System (RTOS) provides?
f) What is a soft timer?
g) Which properties of a system can be checked using formal models like Petri net? Mention two properties.
h) What does preemption of a task mean?
i) What does it mean to say that a resource has “mutually exclusive access”?
j) When does a deadlock happen?

2. (12 points)

We want to write a program that controls the temperature of a room. The program receives the *desired* temperature from a user (for example remotely through internet, using a keyboard, etc.). The program reads the current temperature of the room using a temperature sensor that is connected to the computer. There is a cooler and a heater connected to the computer using actuators. Whenever the room temperature drops at least 2 degrees under the desired temperature the heater has to be turned on. Similarly, when the temperature is raised at least 2 degrees above the desired temperature, the cooler has to be turned on.

Write the program consisting of 3 tasks, t1, t2, and t3. Every 30 seconds task t1 checks if there is a new desired room temperature received from the user and puts it in a global variable. Every 6 seconds task t2 reads the current room temperature from the sensor and puts it in a global variable. Every 5 seconds task t3 compares the current room temperature to the desired temperature. If the current temperature is at least 2 degrees below the desired temperature, t3 turns on the heater. If the temperature is at least 2 degrees above the desired temperature t3 turns on the cooler. If the difference between the current temperature and desired temperature is less than 2, both the cooler and the heater have to be turned off. Any shared variable has to be protected so that at any time only one task can have access to it.

Notice: You can use the following function calls in your implementation:

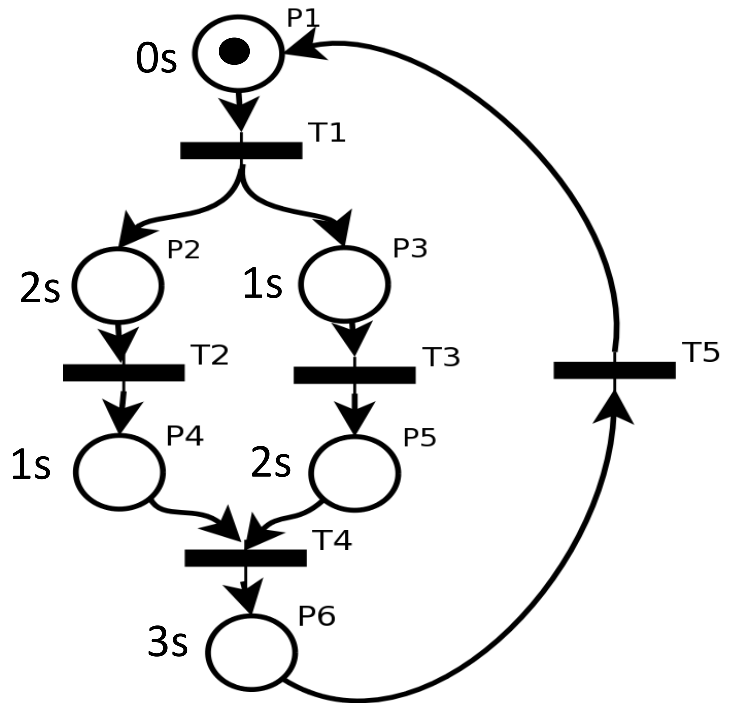
```
float receiveDesired(); // checks and receives the desired room temperature from the user
float readTemperature(); // reads the current room temperature from the sensor
void turnCooler(int i); // turns the cooler on if i==1 and turns it off if i==0
void turnHeater(int i); // turns the heater on if i==1 and turns it off if i==0
void vTaskDelay(pdMS_TO_TICKS(ms)); // the caller task is delayed for ms milliseconds
BaseType_t xTaskCreate( TaskFunction_t pvTaskCode, char * const pcName,
                        configSTACK_DEPTH_TYPE usStackDepth,
                        void *pvParameters,
                        UBaseType_t uxPriority,
                        TaskHandle_t *pxCreatedTask
                      );// creates a task
SemaphoreHandle_t xSemaphoreCreateMutex( void );  // creates a mutex
xSemaphoreTake( SemaphoreHandle_t xSemaphore, TickType_t xTicksToWait ); // locks a mutex
xSemaphoreGive( SemaphoreHandle_t xSemaphore ); // unlocks a mutex
```

3. (5 points)

Assuming the following Petri net executes with maximum speed

a) Draw the reachability graph of the P-timed Petri net

b) Does the Petri net have a period (Is it repeated)? If yes what is the period of the Petri net?



4. (8 points)

a) For a task model where tasks are periodic and preemptive and the tasks are independent, explain briefly how the following scheduling algorithms work. **(2 points)**

   i. Rate Monotonic Scheduling (RMS)

   ii. Earliest Deadline First (EDF)

b) How does Priority Inheritance Protocol (PIP) work? **(2 points)**

c) What is the difference between a Time driven and Event driven scheduling algorithm? **(2 point)**

d) What is the difference between blocking and non-blocking memory management? **(2 point)**

5. (5 points)

The following independent preemptive periodic tasks are to be scheduled by Rate Monotonic Scheduling (RMS) algorithm. The deadline of each task is equal to its period.

a. Calculate the maximum response time only for task3.
b. Is task3 schedulable by RMS?

	e_i (execution time)	p_i(period) = d_i(deadline)
task1	1	3
task2	2	5
task3	3	13

Notice: The following equation is used for calculation of response times:

$$R_i = e_i + \sum_{\tau_j \in H_i} \left\lceil \frac{R_i}{p_j} \right\rceil e_j$$